UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/851,554 | 05/08/2001 | Stepan Sokolov | SUN1P833/P6212 | 4023 |

22434 7590 06/16/2004

BEYER WEAVER & THOMAS LLP
P.O. BOX 778
BERKELEY, CA 94704-0778

| EXAMINER | |
|---|---|
| YIGDALL, MICHAEL J | |
| ART UNIT | PAPER NUMBER |
| 2122 | |

DATE MAILED: 06/16/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/851,554 | SOKOLOV ET AL. | |
| | Examiner | Art Unit | |
| | Michael J. Yigdall | 2122 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *08 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10 September 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *01/03/02, 03/24/03*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. Claims 1-21 are pending and have been examined. The priority date considered for the application is May 8, 2001.

### *Drawings*

2. The drawings are objected to under 37 CFR 1.83(a) because they fail to show the flow of operations 510 and 704 as described in the specification (see the "Yes" and "No" branches from block 510 in FIG. 5, which are not consistent with the description on page 11, line 30 to page 12, line 17; also note that the branches from block 704 in FIG. 7 are not labeled). Any structural detail that is essential for a proper understanding of the disclosed invention should be shown in the drawing. MPEP § 608.02(d). A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

### *Specification*

3. The attempt to incorporate subject matter into this application by reference to the application entitled "Identifying and Tracking Object References in a Java Programming Environment" is improper because the application number has not been provided.

### *Double Patenting*

4. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed.

Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

5. Claims 1-21 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 2-28 and newly added claim 30 (as of the correspondence mailed 6/2/04) of copending Application No. 09/851,663. Although the conflicting claims are not identical, they are not patentably distinct from each other because both recite analogous methods and apparatuses for tracking references to objects in an object-oriented programming environment.

For example, instant claim 1 and conflicting claim 4 (i.e., from App. No. 09/851,663) both recite placing or storing a value or reference on a reference stack associated with an execution stack after determining whether to do so. Instant claim 8 adds to instant claim 1 the additional limitation recited in conflicting claim 4, wherein the value is stored in an entry of the reference stack with the same offset as the offset used with the execution stack. It would be obvious to perform the translating step of instant claim 1 in association with the determining step

of conflicting claim 4 when it may be necessary to interpret, convert, translate, etc., the commands or operations in order to perform the determination. Instant claim 2 and conflicting claim 2 both recite the limitation wherein the object-oriented programming environment is a Java environment, and likewise instant claim 7 and conflicting claim 3 both recite the limitation wherein the reference stack and the execution stack have the same size. Instant claim 4 is analogous to conflicting claims 7-10, which recite the same Getfield, Aload, Getstatic and Areturn operations.

Furthermore, instant claims 9, 15 and 20 are variations of instant claim 1, while conflicting claims 12, 14, 18, 21, 24 and 30 are variations of conflicting claim 4. Therefore, the explanation presented above applies to these claims as well. Additionally, the Java Bytecode verifier recited in instant claim 15 would be a component of the virtual machine recited in conflicting claims 21, 24 and 30, and conflicting claim 25 further recites such Bytecode verification.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

### *Claim Objections*

6. Claim 11 is objected to because of the following informalities: The phrase "wherein said Java command" should perhaps be replaced with either --wherein said determining that said Java command-- (as in claim 10), or --wherein said determining of whether said Java command-- (as in claim 12). Appropriate correction is required.

### *Claim Rejections - 35 USC § 112*

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claim 21 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 21 recites the limitation, "a computer readable medium as recited in claim 1." There is insufficient antecedent basis for this limitation in the claim. Claim 1 does not recite a computer readable medium. Claim 21 was perhaps intended to depend from claim 20.

### *Claim Rejections - 35 USC § 102*

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-4, 7-10, 15, 16, 20 and 21 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Pat. No. 5,903,899 to Steele, Jr. (hereinafter "Steele").

With respect to claim 1, Steele discloses a method of tracking references to objects of an object oriented programming environment (see the title and abstract), said method comprising:

(a) determining whether a command is likely to place a reference to an object on an execution stack of said object-oriented programming environment (see column 8, lines 28-39, which shows determining whether the stack operands and results of an instruction or command are references, i.e. references to objects);

(b) translating said command into another command when said determining determines that said command is likely to place a reference to an object on an execution stack of said object-oriented programming environment (see column 8, lines 28-39, which shows replacing or translating the instruction into another instruction; see also column 7, lines 57-64); and

(c) placing a reference to said object on a reference stack associated with said execution stack when said another command is executed (see column 7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 2, Steele further discloses the limitation wherein said object-oriented programming environment is a Java operating environment (see column 4, lines 62-66).

With respect to claim 3, Steele further discloses the limitation wherein said determining of whether a command is likely to place a reference on said execution stack is performed during Java Bytecode verification (see column 16, lines 10-18, which shows performing the determination when verifying Java bytecode).

With respect to claim 4, Steele further discloses the limitation wherein said determining of whether a command is likely to place a reference on said execution stack operates to determine whether a Getfield, Aload, Getstatic, or Return command is being performed (see column 13, lines 46-56, which shows identifying getfield and getstatic instructions for use with

the reference stack, and column 7, line 65 to column 8, line 15, which shows the same for the aload instruction; see also column 8, lines 28-39, which shows identifying stack results, i.e. return values associated with a return command, as references).

With respect to claim 7, Steele further discloses the limitation wherein said reference stack and said execution stack have the same size (see column 4, lines 23-29, which shows that the sub-stacks used for distinguishing references occupy the same memory as the single stack, i.e. the execution stack; note that such arrangement enables the reference stack and the execution stack to have the same size).

With respect to claim 8, Steele further discloses the limitation wherein at least one reference to an object is stored in an entry with an offset that is the same as the offset used to store said at least one reference in said execution stack, when said another command is executed (see column 14, lines 21-46, which shows that the references are stored in slots or entries and that the offsets are numbered such that the ordering of the offsets is the same as the original).

With respect to claim 9, Steele discloses a method of tracking references to Java objects in a Java programming environment (see the title, abstract and column 4, lines 62-66), said method comprising:

(a) determining whether said Java command is likely to place the only reference to a Java object on the execution stack (see column 8, lines 28-39, which shows determining whether the stack operands and results of an instruction or command are references, i.e. references to objects; see also column 2, lines 6-17, which shows identifying reachable objects, i.e. determining if there are references to an object or if a reference to an object is the only reference to the object);

(b) translating said command into another command when said determining determines that said Java command is likely to place the only reference to a Java object on the execution stack (see column 8, lines 28-39, which shows replacing or translating the instruction into another instruction; see also column 7, lines 57-64);

(c) executing said Java command (note that the instructions or commands are inherently executed; see also column 2, lines 6-17, which shows operating with an executing program);

(d) placing a reference to said object on a reference stack associated with said execution stack when said another command is executed (see column 7, lines 33-56, which shows placing references on a reference stack during execution); and

(e) wherein said determining of whether said Java command is likely to place the only reference to a Java object on the execution stack is performed during Java Bytecode verification (see column 16, lines 10-18, which shows performing the determination when verifying Java bytecode).

With respect to claim 10, Steele further discloses the limitation wherein said determining that said Java command is likely to place the only reference to a Java object on the execution stack further comprises determining whether a Getfield, Aload, Getstatic, or Areturn command is being performed (see column 13, lines 46-56, which shows identifying getfield and getstatic instructions for use with the reference stack, and column 7, line 65 to column 8, line 15, which shows the same for the aload instruction; see also column 8, lines 28-39, which shows identifying stack results, i.e. return values associated with an areturn command, as references).

With respect to claim 15, Steele discloses a Java Bytecode verifier suitable for operating in a Java operating environment (see the title, abstract and column 4, lines 62-66; see also column 16, lines 10-18, which shows a Java bytecode verifier),

(a) wherein said Bytecode verifier operates to determine whether there is at least one Java command in a stream of Java Bytecode commands such that said at least one Java command is likely to place the only reference to a Java object on the execution stack (see column 8, lines 28-39, which shows determining whether the stack operands and results of an instruction or command are references, i.e. references to objects; see also column 2, lines 6-17, which shows identifying reachable objects, i.e. determining if there are references to an object or if a reference to an object is the only reference to the object);

(b) wherein said Bytecode verifier operates to translate said Java command into another Java command when said Java command is likely to place the only reference to a Java object on the execution stack (see column 8, lines 28-39, which shows replacing or translating the instruction into another instruction; see also column 7, lines 57-64); and

(c) wherein a reference associated with said command is placed on a reference stack as well as said execution stack when said another command is executed (see column 7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 16, Steele further discloses the limitation wherein said Bytecode verifier operates to determine whether a Getfield, Aload, Getstatic, or Areturn command is being performed (see column 13, lines 46-56, which shows identifying getfield and getstatic instructions for use with the reference stack, and column 7, line 65 to column 8, line 15, which

shows the same for the aload instruction; see also column 8, lines 28-39, which shows identifying stack results, i.e. return values associated with an areturn command, as references).

With respect claim 20, Steele discloses a computer readable medium including computer program code for tracking references to objects of an object-oriented programming environment (see the title, abstract and column 17, lines 20-57), said computer readable medium comprising:

(a) computer program code for determining whether a command is likely to place a reference to an object on an execution stack of said object-oriented programming environment (see column 8, lines 28-39, which shows determining whether the stack operands and results of an instruction or command are references, i.e. references to objects);

(b) computer program code for translating said command into another command when said determining determines that said command is likely to place a reference to an object on an execution stack of said object-oriented programming environment (see column 8, lines 28-39, which shows replacing or translating the instruction into another instruction; see also column 7, lines 57-64); and

(c) computer program code for placing a reference to said object on a reference stack associated with said execution stack when said another command is executed (see column 7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 21, Steele further discloses the limitation wherein said object-oriented programming environment is a Java operating environment (see column 4, lines 62-66).

***Claim Rejections - 35 USC § 103***

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 5, 6, 11-14 and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Steele, as applied to claims 4, 10 and 16 above, respectively, in view of U.S. Pat. No. 6,047,125 to Agesen et al. (hereinafter "Agesen").

With respect to claim 5, although Steele discloses placing an address or reference on the stack due to a method invocation, i.e. a change in the control flow, Steele does not expressly disclose the limitation wherein said determining of whether a command is likely to place a reference on said execution stack further comprises determining whether there is a change in the flow control.

However, Agesen discloses identifying a plurality of control flow paths and determining whether a variable represents a reference, i.e. whether a command is likely to place a reference on the stack, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the control flow determination features taught by Agesen with the reference tracking system of Steele for the purpose of further improving garbage collection.

With respect to claim 6, although Steele discloses identifying putfield instructions for use with the reference stack (see column 13, lines 46-56), Steele does not expressly disclose the limitation wherein said determining of whether a command is likely to place a reference on said execution stack further comprises determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

However, Agesen discloses determining whether a plurality of control flow paths use the same variable for both a reference and a primitive value, i.e. whether a command is likely to overwrite a reference with a primitive value or vice versa, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the overwrite determination features taught by Agesen with the reference tracking system of Steele, which identifies putfield instructions, for the purpose of further improving garbage collection.

With respect to claim 11, although Steele discloses placing an address or reference on the stack due to a method invocation, i.e. a change in the control flow, Steele does not expressly disclose the limitation wherein said Java command is likely to place the only reference to a Java object on the execution stack further comprises determining whether there is a change in the flow control.

However, Agesen discloses identifying a plurality of control flow paths and determining whether a variable represents a reference, i.e. whether a command is likely to place a reference

on the stack, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the control flow determination features taught by Agesen with the reference tracking system of Steele for the purpose of further improving garbage collection.

With respect to claim 12, although Steele discloses identifying putfield instructions for use with the reference stack (see column 13, lines 46-56), Steele does not expressly disclose the limitation wherein said determining of whether said Java command is likely to place the only reference to a Java object on the execution stack further comprises determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

However, Agesen discloses determining whether a plurality of control flow paths use the same variable for both a reference and a primitive value, i.e. whether a command is likely to overwrite a reference with a primitive value or vice versa, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the overwrite determination features taught by Agesen with the reference tracking system of Steele, which identifies putfield instructions, for the purpose of further improving garbage collection.

With respect to claim 13, Steele further discloses the limitation wherein said reference stack and said execution stack have the same size (see column 4, lines 23-29, which shows that

the sub-stacks used for distinguishing references occupy the same memory as the single stack, i.e. the execution stack; note that such arrangement enables the reference stack and the execution stack to have the same size).

With respect to claim 14, Steele further discloses the limitation wherein at least one reference to a Java object is stored in an entry with an offset that is the same as the offset used to store said at least one reference in said execution stack, when said another command is executed (see column 14, lines 21-46, which shows that the references are stored in slots or entries and that the offsets are numbered such that the ordering of the offsets is the same as the original).

With respect to claim 17, although Steele discloses placing an address or reference on the stack due to a method invocation, i.e. a change in the control flow, Steele does not expressly disclose the limitation wherein said Bytecode verifier operates to determine whether there is a change in the flow control.

However, Agesen discloses identifying a plurality of control flow paths and determining whether a variable represents a reference, i.e. whether a command is likely to place a reference on the stack, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the control flow determination features taught by Agesen with the reference tracking system of Steele for the purpose of further improving garbage collection.

With respect to claim 18, although Steele discloses identifying putfield instructions for use with the reference stack (see column 13, lines 46-56), Steele does not expressly disclose the

limitation wherein said Bytecode verifier operates to determine whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used.

However, Agesen discloses determining whether a plurality of control flow paths use the same variable for both a reference and a primitive value, i.e. whether a command is likely to overwrite a reference with a primitive value or vice versa, in a system for eliminating variable conflicts and thereby improving garbage collection (see column 5, lines 16-27 and 48-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the overwrite determination features taught by Agesen with the reference tracking system of Steele, which identifies putfield instructions, for the purpose of further improving garbage collection.

With respect to claim 19, the limitations recited in this claim are identical to the limitations recited in claims 17 and 18. Therefore, see the explanations for claims 17 and 18 presented above.

13. Furthermore, the examiner would like to direct Applicant's attention to International Pub. No. WO 99/10811 (art of record; document B2 from the IDS filed 3/24/03), which discloses features of the invention substantially as recited in the claims. For example, WO 99/10811 shows a system for tracking references to objects (see page 2, line 18 to page 3, line 1) in a Java environment (see page 6, lines 10-17) having a reference stack associated with each stack frame or execution stack to identify the only reference to an object (see page 6, lines 18-29), wherein both stacks may have same size and object references are placed on the reference stack when it is determined that they are placed on the execution stack (see page 8, lines 4-18).

***Conclusion***

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. U.S. Pat. No. 6,093,216 to Adl-Tabatabai et al. discloses a method for tracking object references in Java programs at runtime. U.S. Pat. No. 6,304,949 to Houlsdworth discloses a method of garbage collection comprising a reference stack.

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (703) 305-0352. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER